Detecting Anonymising Proxy Usage on the Internet

Wireless Pers Commun
DOI 10.1007/s11277-013-1451-y

Detecting Anonymising Proxy Usage on the Internet

Ronan O'Flaherty · Kevin Curran

Abstract Anonymising proxies are a growing problem for organisations as more people become aware of their capabilities. These proxy sites enable users to bypass the network's filtering system leaving the network open to banned content and harmful threats. Network administrators do try to block these online proxy sites, but with a growing number of new sites created, this task is not a trivial one. Many existing solutions rely almost entirely on Access Control Lists, which blacklist undesirable websites; the end result being that many users learn that anonymous proxies allow them to easily bypass this filtering. While Access Control Lists serve a purpose, there are lots of difficulties especially in detecting if users are circumventing the policies and Access Control Lists. One possible solution is to focus on detecting access to anonymous proxies. This paper outlines procedures required to discover anonymous proxies on the Internet.

Keywords Security · Network security · Proxies · WWW

1 Introduction

In today's world, technology is commonplace and has helped the world to become better connected. Many of these technological advances are beneficial however there is a lot of scope for the technology to be misused. Daily use of the Internet continues to grow and over the last few years the increased popularity of social media sites such as Facebook, Twitter and YouTube, has led to some detrimental effects on the working productivity of its users. As a result the need for institutions to block access to such sites has grown. Existing measures to combat this issue have encouraged users to try and find alternative means to access these

R. O'Flaherty · K. Curran (✉)
School of Computing and Intelligent Systems, Faculty of Computing and Engineering,
University of Ulster, Londonderry, UK
e-mail: kj.curran@ulster.ac.uk

R. O'Flaherty
e-mail: oflaherty-r1@email.ulster.ac.uk

Published online: 19 October 2013

and other sites. Aiding the problem is the vast amount of sites and programs freely available which can be used to bypass the security measures in place and provide access to sites and content that should not be available to these users.

Early attempts by network administrators were to block access to the websites by denying the IP address of a range of websites. For a short time, this method worked in keeping network users from accessing their favourite social sites from within their security enabled websites; but now proxy websites can be used to view outside content. A proxy is a web-based script that will allow anyone who is accessing it to view any website, as it masks the real IP address of the source and fools the system into allowing the blocked website to be viewed. This has become more troublesome as these PHP/CGI web scripts are freely available on the Internet to download, and users can setup their own proxy site very easily, if their preferred choice is ever detected and blocked.

Another option open to network users to bypass the security measures is to install an onion routing browser application, which will use a different protocol layer to route the Internet traffic through a different port than is standard (80 or 443) and allow access to blocked content that way. In the same that way a website leaves a fingerprint when it has been accessed [1], when a proxy script or onion routing application has been accessed this should also leave its own fingerprint. This paper will detail how proxy and onion routing browsing use on a network can be detected on the client-side and used to help block access to offending websites, if some of the more traditional security measures have failed.

2 Existing Approaches to Detecting Proxies

Here we look at the processes used by network administrators to secure their network systems, while also looking at procedures used to bypass these measures.

2.1 IP Blocking

IP blocking is the most basic technique used to combat potential threats to networks [2]. The network administrator can block an IP address (or range of IP addresses) from accessing certain domain name IP addresses. This prevents a direct connection being made between the local machine(s) within the given IP address range and the server of the blocked domain [3]. This can also work vice versa, where the website administrators of a site can block access to a disruptive user by obtaining their IP address and blocking access to the site through the administration panel for the domain name [4]. While noted as one of the more basic methods of denying access to certain websites, it is also a very effective method, as it guarantees that the site is blocked, enabling a safer network for its users.

2.2 Access Control Lists

Access Control Lists (ACL) are usually implemented along-side IP blocking techniques to help secure a network [5]. They work by creating a list of accessible ports on the local machine linked with what the ports it should be opening. Applications that run and their associated ports are also noted. Any port not mentioned in the ACL will be considered 'locked down' and not usable. Security is crucial for an ACL to function correctly [6], so when a request is received to open and use a port, the ACL is checked, firstly to see if the port is allowed to be opened and secondly, if the application requesting the opening of the port is actually allowed to use that port. If the two criteria are met, then the port will be opened for this functionality;

if they are not met, then the port will remain blocked. The main downside to ACLs is that genuine applications attempting to open ports might not get access if they are not included in the list. In a case like this, a request must be sent to the network administrator to add the port and application to the list. This can take up valuable time removing the administrator from more important duties. Another downside is that the implementation of an ACL on a larger scale can also prove troublesome [7].

2.3 Base64 Encoding

Base64 encoding is a form of encoding data into an ASCII string of characters, so that it can be securely and safely sent along a transmission line that deals primarily with text [8]. This can be used within a web based PHP program to obfuscate the code of the program, so that it is less understandable by humans. This means that it can be used maliciously to slip code past security measures and potentially leaves networks vulnerable to attack, as it will be executing commands that it cannot understand because of the base64 encoding [9].

Base64 takes each character, translates it to its ASCII value, stores the ASCII value in binary, 8-bit format as 1s and 0s, and then grabs 6 bit groups, these 6 bits are required to represent all possible Base64 values. Base64 values are always expressed in 4-byte groupings (three ASCII characters become four Base64 characters) so a Base64 value should always be divisible by four. If the original string does not take up the entire four bytes, the Base 64 value is supposed to be padded with the '=' sign character, which represents a null value.

The disadvantage of base64 encoding of data is that it is complicated and time consuming to encode and thus prone to mistakes. Despite possible misuse, there are legitimate reasons for including it, such as encoding copyright notices into free website templates, so that most users will not know how to decrypt the code that is in place, without massive changes to the design of the template.

2.4 Onion Routing

Another method used to bypass a network's security measures is using an onion routing browser. This is an application that runs like any web browser, but offers its users the chance to surf the Internet anonymously and not tying the user down to one single IP address [10]. This type of application works in two steps. The first is routing the web traffic to the Transmission Control Protocol (TCP) over the default HTTP—this is where the onion part of the title comes from. It allows the user to browse the Internet using a different layer in the TCP/IP stack. The second step is that the application opens a different port to allow traffic in and out of the machine. Normal Internet browsing uses port 80 (or 443 if it is secure connection), but an onion routing application will open any one of a range of ports from 222, 9001–9004, 9030–9033 and 9100 [11]. This is where the second part of its name comes from, in that the application will route Internet traffic to a different port. To make the use of the application more secure, at some stage it will use the 'TLSv1' protocol as an alternative to opening the 443 port. To fully understand the workings of onion routings, it is necessary to analyse how each section of the program plays its own part to allowing browsing [12]. A major advantage of using onion routing applications, is that it that it allows its users to browse the Internet not just securely, but also anonymously, so it does not tie them down to one IP address and can potentially open content that would not necessarily be available to them. Reed's paper [13] provides a basic overview of how proxies and onion routing applications work together.

2.5 Common Programs

Despite a raft of these measures currently in place on network systems, there are people who feel that they should still be allowed to have access to websites that their network administrator has deemed unsuitable for that network. The practice of bypassing these security measures is now so widespread that organisations have been forced to ask those using their networks to sign contracts stating that they will not attempt to counteract any security measures on the network before they are even allowed onto the system. Web-based proxy servers have become one of two standard methods for countering security enabled networks. These are PHP or CGI website scripts that are readily available to downloadable from the Internet. All that is needed to obtain the script is a quick search using one of the many search engines. These scripts can be downloaded and installed on any web server (as long as it allows dynamic scripts to be installed and executed on it). Two popular PHP backed web-based proxy services to download are "PHP Proxy" [14] and "Glype" [15]. There is also a more secure proxy script which can be downloaded which is CGI backed called "CGI Proxy" [16]. All of these web-based scripts can be downloaded and setup at home, then used the next day on the security enabled network. Then, if the proxy is detected and blocked, the script can be easily moved to a new server, which is not blocked on the network.

For onion routing applications, the most common program of this nature is the "TOR Browser" [17] and like the proxy scripts, it can be found after a simple Internet search. The "TOR Browser" works like any other internet browsing application ("Internet Explorer", "Firefox") and the executable file can be downloaded and installed on the hard drive and used to bypass security measures that are in place. Some attempts recently to block the use of such a program have been network administrations not allowing users to install programs on their machines without prior consent, and the blocking of USB ports on the local machine so that a portable version could not be installed on a USB stick and plugged in to allow access. Compounding these reasons, the "TOR Browser" appears to work quite slowly (Fig. 1).

3 Proxy Detection System Design

This section outlines more aspects of the Intrusion Detection System (IDS) program (see Fig. 2), which runs on the client to detect the use of proxy websites or onion routing browsers.

It is important for network administrators to monitor for proxy and onion routing application use as these scripts and applications can be used to bypass security measures that are in place on a network. This poses additional risks to the network and that is why most network administrators will want to know if this is happened. Naturally, there are some more legitimate reasons why someone could use a proxy site [18], including masking their actual location if checking out a competitors website or to filter certain types of web scripts from running. The main concern is that if someone is using a web-based proxy script, there is a high possibility that they are not using it for legitimate means.

Initially, the system 'sniffs' the network traffic. The only way to accomplish this on the client-side of a network is to detect proxy or onion routing browsing at the source. Programs such as "Wireshark" [19], Httpfox (see Fig. 1) or "Snort" [20] can be used to monitor the traffic movement on a network in real-time. A useful feature of these programs is their ability to log the detected traffic movement to a text (.txt) file. For the creation of the IDS program, this log file will become the output for the first stage of the detection.

The cornerstone of the program is the log file produced from the network monitoring platform. This will be "read" into a detection program—which is the second stage of the

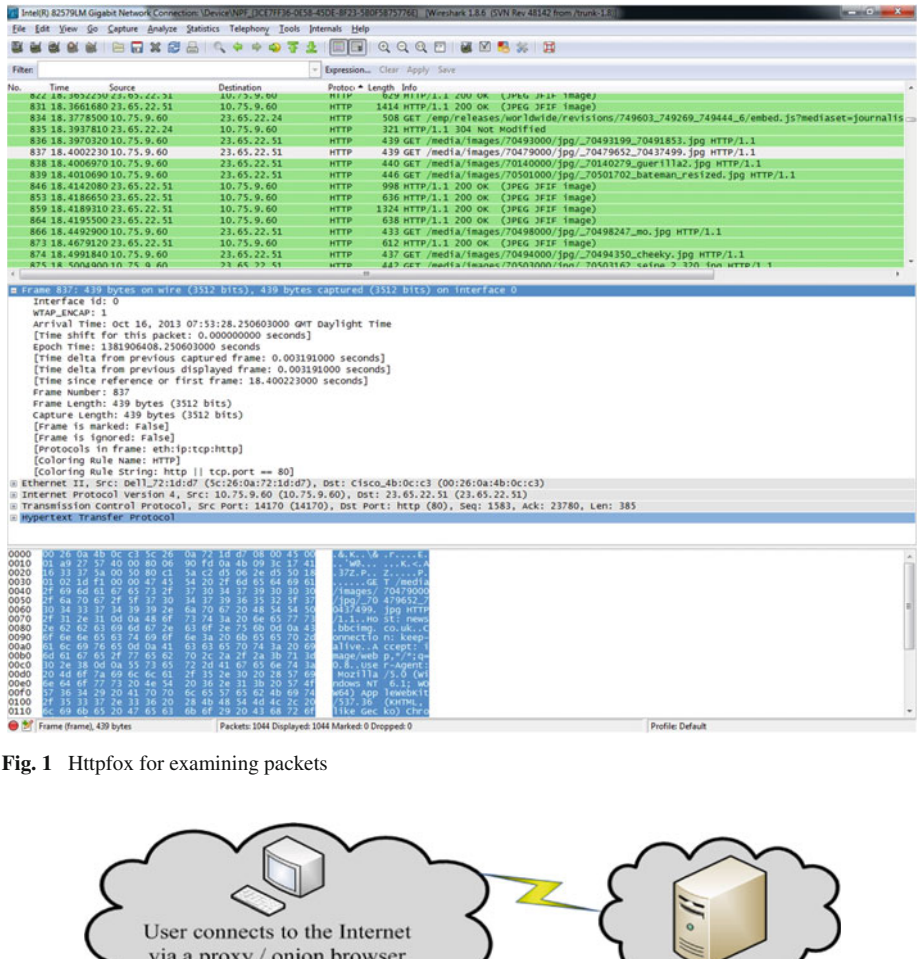


Fig. 1 Httpfox for examining packets

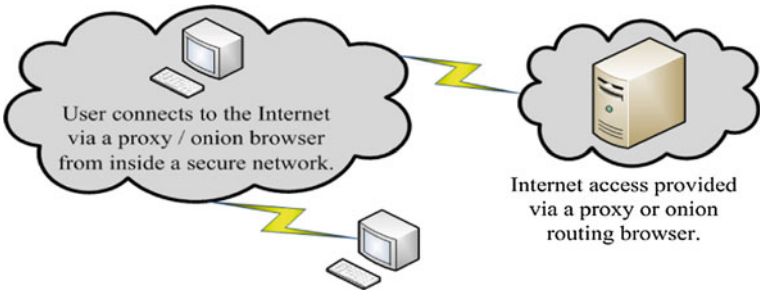


Fig. 2 System architecture

design. The IDS program will be a command line based C-program which will run by scanning the text log file line-by-line. As the program reads through the file, a series of pre-defined characteristic strings will be matched against the file being read in. These characteristic strings will be unique to the various web based proxy sites and onion routing browsing traffic. As each characteristic string set is found, this will toggle a flag within the program.

The final stage of the IDS program is the rule-base. This produces an on-screen notification as to the outcome of the analysis of the traffic monitoring output file. The rule base will be setup with several rules to match the flags. As the program will be used to detect the use of multiple systems at any one time, potentially a few of the flags could be toggled during one file. The rule-base within the program checks to see if any of the detection flags have been

toggled. The rule-base will be setup to determine if flag 1, flag 3 and flag 5 for example, have all been toggled, then this will indicate the use of a web-based proxy script. Various rules are put in place and if any series of toggles have been met, a message will be displayed on the command line indicating that a proxy or onion routing browsing use has been detected, or if no such use is suspected. The network administrator will be able to see that proxy or onion routing browsing has been detected, track down its use and prevent future misuse of their system. Ideally, the time taken from the log file being read into the intrusion detection system, to the decision output being displayed on the screen should only be a few seconds. To catch and trap proxy or onion routing use on a network, the key is analysing and comparing to 'normal' traffic flows.

Even though the proxy scripts and onion routing applications are described as anonymous, suggesting that it creates an invisible user who can access the web, any scripts when executed, will have their tell-tale signs of execution and for a web-based proxy (or onion routing browser) this is no different. When one of these scripts or programs makes a user anonymous; this masks the location and the identity of the computer from the website it is attempting to access. This is why the network traffic is monitored to determine the various tell-tale signs of the proxy scripts in action as they are being run client-side, no matter what the developer attempts to do; there is no way this can be masked.

4 Proxy Detection System Implementation

Before any part of the intrusion detection system can be developed, there is some software which has to be installed. Firstly, the program for analysing and monitoring network traffic "Wireshark" has to be installed; this can be obtained from their website [19]. The Windows installer for this also includes the "WinPcap" program, which enables the packet analyzing functionality. Next, the first of the anonymous browsing programs should also be installed. The "TOR Browser" can be downloaded from its official website [17]. These programs should be tested to make sure they function as expected before attempting to log any of the network traffic for analysis. The web scripts also need to be implemented on the server. The three different scripts which were used for testing were "PHP Proxy" [14], "Glype" [15] and "CGI Proxy" [16]. These are available as ZIP files to download. Before files are uploaded to a server, PHP and CGI functionality must be enabled. An Internet browser can be used to access the scripts directly and perform any installation procedure that needs to be undertaken. When this has been completed, the web scripts can be used to anonymously browse the Internet.

To monitor the movement of traffic on the client-side, the network protocol analysing program "Wireshark" is used. "Wireshark" listens to the network and logs the packets that it captures to an output file. For practicality and to reduce the size of the log file that will be fed to the IDS program, "Wireshark" saves a packet summary line and any packet details displayed on the live capture window in a log file (see Fig. 3). A database of web-based proxy scripts for testing was created. This included "PHP Proxy", "Glype" and "CGI Proxy", and the onion routing application, the "TOR Browser". For each section, two log files were created to help discover a pattern of strings common to each proxy site.

Analysing the logs for the first proxy site (PHP Proxy), it was noted that when this proxy service was executed, the "*HTTP*" protocol was used. At the same time, a "*GET*" command was issued on the proxy server; each time this web based proxy was used the "*GET*" command had an attached URL, which contained the string "*index.php?q=aHR0c*". As the index file can be called anything—as long as it is suffixed with '.php'—this was removed from the detection string (Table 1).

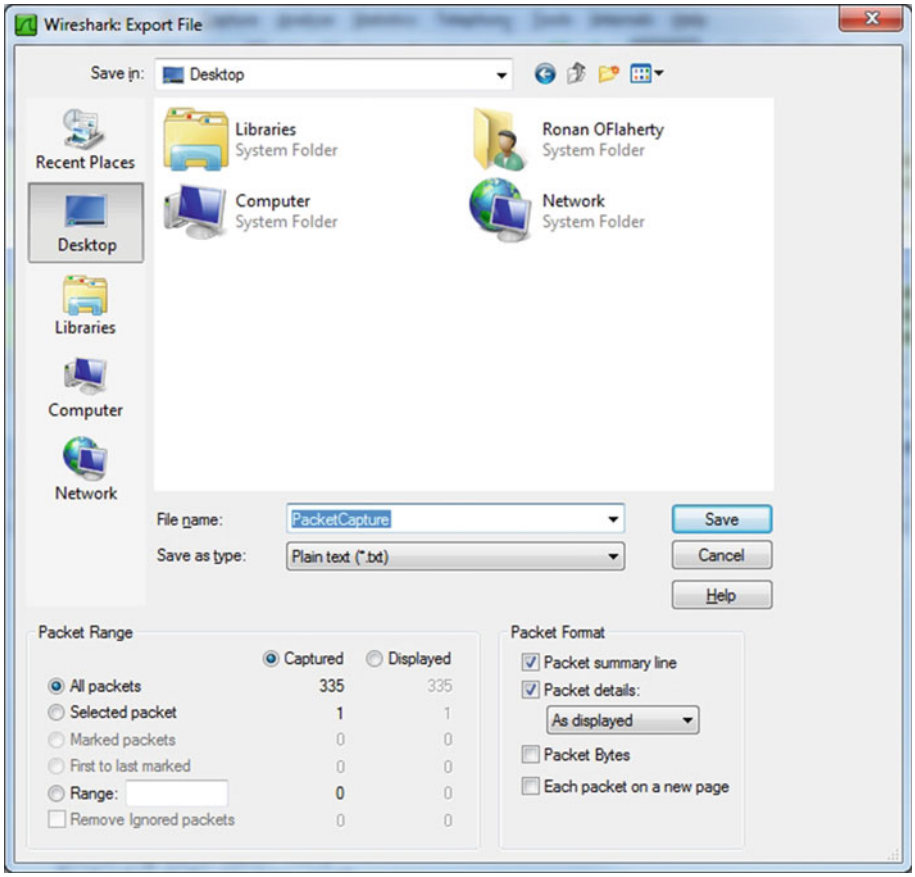


Fig. 3 Capturing packets

Table 1 Proxy One—PHP proxy characteristics

Protocol	HTTP
Command	GET
Detection String	.php?q=aHR0c

Coding the IDS program, three new string values where added, set with the three characteristic values outlined above. Each variable also has its own flag, so that when the string is found the flag can be toggled to reflect the found value. For the output of the program, an "IF statement" is initialized, so that if the flag for each of the three characteristic values has been toggled, then a message will be displayed warning that proxy use was detected. This rule set was then tested again; the log files for proxy one use and no proxy use—with a 100 % success rate. The message displayed on the terminal screen is shown in Fig. 4.

For the second proxy site (Glype), a thorough investigation was started to find a series of patterns recurring through the two test log files when this proxy was in use. It was discovered that in a similar way to the first proxy, the "*HTTP*" protocol was called, along with the "*GET*" command. The difference for this proxy was that as part of the "*GET*" command, there was a

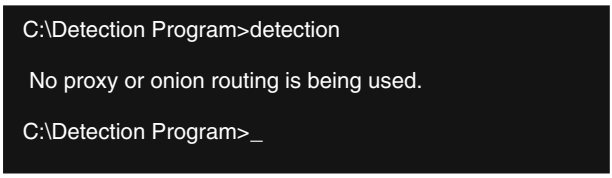


Fig. 4 No proxy detected message

Table 2 Proxy 2—Glype characteristics

Protocol	HTTP
Command	GET
Detection String	.php?u=

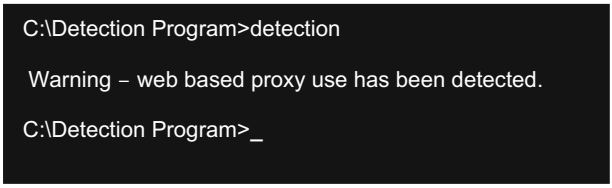


Fig. 5 Proxy detected message

Table 3 Proxy 3—CGI proxy characteristics

Port	443
Protocol	TLSv1
Command	GET
Detection String	.cgi/

slightly different URL attached to it: "*browse.php?u=*" (see Table 2). As happened in the first case, the 'browse' filename can be changed to anything and still function, so the detection string has this removed.

The search strings for the protocol and command are already in place with their respective flags in the program, so a second detection string was added to the program. As before, the rule base was established that if the flags for detection of the protocol and command were toggled, along with the new detection string, then a web-based proxy site was being used. If all the flags are detected, the message shown in Fig. 5 will be displayed. The updated IDS program was tested using the two test log files for the second proxy site and the log files showing that no proxy was being used and again, there was a 100 % success rate.

The third proxy site (CGI Proxy) differs from the previous two in that it was a CGI script, which utilises the secure sockets layer (SSL) protocol. For this web-based proxy, it was discovered that there was four strings that were required to be matched to denote its use. Firstly, as this is a SSL proxy, the secure web port of the computer would have to be opened, "*(443)*". The transport layer security "*TLSv1*" protocol is also applied when this proxy is executed and, just like in the first two proxy sites, the "*GET*" command is called, but this time the URL attached contains the string "*.cgi/*" (see Table 3).

Building on the IDS program structure, three new search strings have to be added to the program—the port number, the new protocol and the detection string—along with flags

Table 4 TOR browser characteristics

Port	222, 9100, 9001–9004, 9030–9033
Protocol	TCP

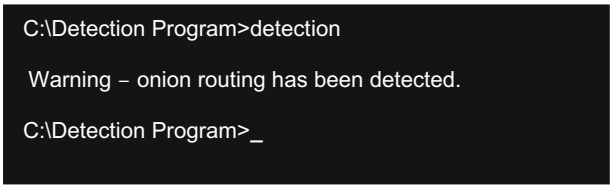


Fig. 6 Proxy detected message

which are toggled if the respective strings that have been found. The warning if this proxy is in use stems from a rule base near the end of the program which queries if all the four flags are toggled. This was initially tested on the two test scenario log files and the two logs of when no proxy was being used—there was a 100 % success for this program. So far, the IDS program has been used to detect web-based proxy scripts. An onion routing browser is a program that works in a similar way to any Internet browser, but offers the user the chance to browse anonymously. The "TOR Browser" works by routing the Internet browsing from where it usually works. Analysing the two log files for the "TOR Browser" identified that the "*TCP*" protocol is used for web browsing and then the TOR browser opens up one of ten different ports (*222*, *9001* to *9004*, *9030* to *9033* and *9100*). The port numbers opened are a set standard for onion routing browsers and would not usually be used for standard Internet browsing (Table 4).

Onion routing browsing is detected by searching for the protocol string and matching it with one of the ten possible ports that could be opened. Once again, detection flags are initialised and toggled if the protocol and one of the port strings have been found; if they are found, a warning message is printed to the screen to indicate that onion routing use has been detected. The final part of the IDS program was tested with the two TOR log files and the two log files for normal Internet use it provided a 100 % success rate. Once again, a message is displayed on the screen, as shown in Fig. 6, to warn if onion routing use has been detected from the log file.

To complete the implementation, the entire program was tested using the eight example log files. Each of the log files were run through the IDS program and the program's output was checked to ensure that it printed the correct message to the screen—discovering if there were any bugs in the system. As expected, all eight files were processed by the IDS program and displayed their correct outputs printed on the screen. A more rigorous test of the system was now able to be carried out.

5 Proxy Detection System Testing

The system was tested to verify its effectiveness at detecting proxy or onion routing browsing. Testing was carried out on a number of 'Normal' Internet activities (see Table 5).

Table 5 Testing scenarios

Test	Activity
01	Browsing and shopping on Amazon
02	Logging in and using Facebook
03	Searching for and watching a YouTube video
04	Logging in and using Twitter
05	Browsing and watching videos on BBC news
06	Making a post on a message board
07	Logging in and sending an e-mail
08	Windows Media stream of a radio station
09	Search Google and using Wikipedia
10	Locating and downloading a ZIP file

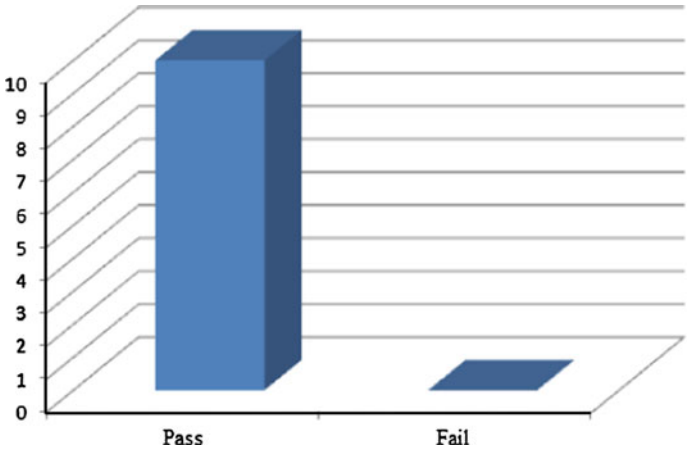


Fig. 7 IDS program's results for when no proxy site was used

Testing was carried out from the home page of the browser for the "TOR Browser" and from the execution page of one of the web-based proxy sites. There was also a control experiment where no proxy or onion routing application was used. In total, 50 log files were created to test the IDS program.

5.1 Expected Outcomes

To begin the testing phase, a log was taken for each of the ten activities, with no proxy site or onion routing browser being used to perform the activities (see Fig. 7). The traffic was monitored using "Wireshark" and the log file for each was saved. Individually, each file was read into the IDS program and tested against the rule base established in the implementation section. For each of the ten cases, the message "No proxy or onion routing is in use" appeared on the screen, indicating a 100 % success rate for the first batch of testing. This is the expected outcome.

With confirmation that the initial testing of the system worked and correctly determined when no proxy script or onion routing application was being used, testing continued using the first proxy website (PHP Proxy).

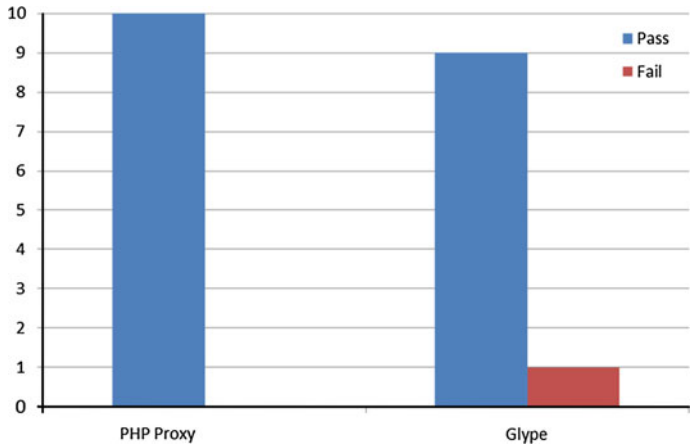


Fig. 8 Pass and fail rates for the tests performed using PHP and Glype

"Wireshark" was again used to log each of the ten scenarios and these log files were then analysed by the program. For the ten cases tested while using the web based proxy, all ten were correctly identified by the program as having a proxy in use, with the following message displayed on the screen, "Warning—web based proxy use has been detected".

So far, the testing has proven to be very accurate—with no fails in the first twenty testing cases. It was now the turn of the second proxy site (Glype) and, like the previous tests, here ten log files were created from a "Wireshark" capture of the proxy in use. When run through the IDS program, all but one of the log files was identified as using a proxy. The test that failed to show up as using a proxy was the sixth test scenario—posting on a message board. Having analysed the log file by hand, it is believed that the "Wireshark "trace began a few seconds too late, as it was completing another process and it did not pick up the web-based proxy beginning to run. This reduced the success rate of the IDS program for this batch of tests down to 90 %, shown in Fig. 8. A retest of this case was later performed, and the IDS program detected that a proxy was in use.

The testing continued with the third proxy site (CGI Proxy), which styles itself as a proxy that uses the SSL layer to be more secure. The ten activities were logged by "Wireshark" and the output text files analysed by the IDS program. For each file processed by the program, a message to inform the user that a proxy was being used was shown on the screen, providing this batch of test log files with a 100 % success rate, as shown in the green graph in Fig. 9.

The final tests were carried out with "Wireshark" monitoring the use the "TOR Browser" to undertake the ten tasks. The onion routing browser is self-advertised as useable to "defend against a form of network surveillance that threatens personal freedom and privacy" and so makes the user anonymous while the browser runs. From the ten initial activities monitored, two of the test cases failed to be detected by the IDS program, signifying an 80 % success rate for the program.

Bringing all the testing together provides a very telling picture of the accuracy of the IDS program that was created (see Fig. 10). Of the five batches of log files examined by the system, which was 50 log files in total, three batches were identified fully correct for their type of Internet use, while of the other two batches the lowest success rate obtained was 80 %. Overall, this means that the intrusion detection system created has a healthy 94 % success rate, with an average of only 6 % of analysed files being incorrectly identified, therefore slipping through the security net.

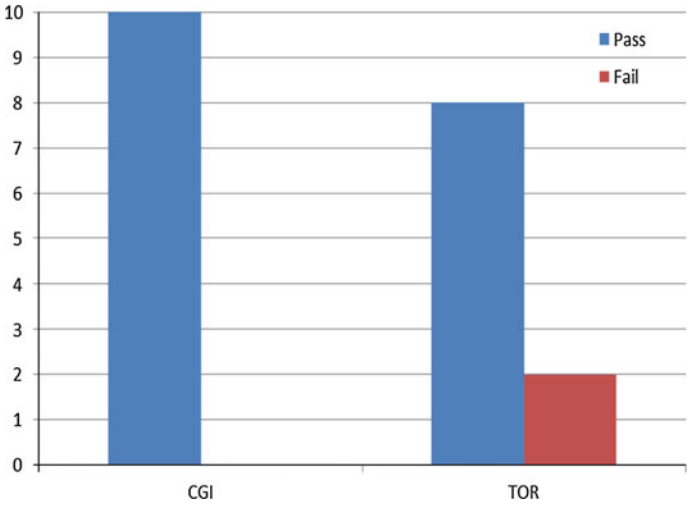
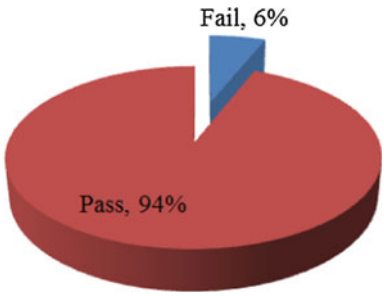


Fig. 9 Pass and fail rates for the tests performed using CGI and TOR

Fig. 10 Overall performance result of the IDS program



5.2 Unexpected Outcomes

Until now, all the testing of the IDS program has been performed on tasks where the outcome has been expected. These tests have been carried out under very controlled circumstances, where the IDS program has been designed specifically to identify the fingerprints from the use of the three web based scripts and the onion browser. During the earlier testing we have seen how some of the "TOR Browser" activity was misidentified by the IDS program and provided a false negative outcome. These unexpected tests are designed to attempt to see if the system can be broken and how it handles the execution of scripts that it has not been specifically designed to identify. The system was then tested to see if there were any other false positive or false negative decisions that can be produced.

A test of the program was carried out using a secure HTTP connection. Several tests were run including purchasing a MP3 file from Amazon, downloading the file from their secure server and sending and receiving e-mails from the secure university e-mail system. The reason behind these tests was that the computer would be forced to open the secure port (443) and use the "TLSv1" protocol; so this might trigger the system to have detected the standard and acceptable Internet use as unwanted use. When the log files for this Internet use were run through the detection system program, it correctly identified these logs as normal Internet access.

Another test utilising a secure Internet connection was using a Secure Socket Layer (SSL) proxy sites, other than the "CGI Proxy" that the IDS program was implemented with and designed specifically to locate its character strings. There were three separate tests of SSL proxy sites, all using web-based SSL scripts. Theoretically, a SSL proxy should use the "*TCP*" protocol and possibly the port 443—but SSL is unique in that it can secure any port that it opens [21]. When an analysis of the log files was completed by the system, all three were incorrectly identified as not using a web-based proxy script. This means that if a user was to execute these scripts on their machine, then the IDS program would not detect its use, and deem this acceptable Internet use. Closer inspection of the log file for the SSL proxies shows that they only just missed out on a correct classification, using the characteristics determined for "CGI Proxy" (see Table 3). The log files show that the port 443 was opened, along with the "TLSv1" protocol (although this can be changed to "TCP" to match up with how the proxy should work) and the "*GET*" command was called. Where the identification failed, no string containing "*.cgi/*" was found in the log details attached to the "*GET*" command. The "CGI Proxy" detection string was not discovered, but for these types of web-based proxies the detection string of "ssl" was attached to the "*GET*" command in two of the three log files. This means that with a small edit to the IDS program, the detection success rate for this type of proxy script greatly increases. An additional, unexpected test was run on the IDS program for the use of the voice over IP messaging and the social program "Skype". A "Skype" conversation is secure between the two or more connections involved in a conversation. An analysis of the log file correctly detects no proxy or onion routing use from the traffic monitor. A closer inspection of the log reveals that a secure port 443 was opened, along with the "*TLSv1*" protocol; there was even a "*GET*" command but the detection string was not found. "Skype" uses the UDP protocol to allow the secure connection between the running Skype applications [22], so the IDS program's rule base would not misidentify it.

To examine the rule base for the "TOR Browser" more closely, a test had to be performed that would require the use of the "*TCP*" protocol. For this test, a simple JPEG image file was uploaded to a webserver and downloaded again from the server. The "*TCP*" protocol would be needed to make the connection and there was a possibility that it could open one of the ten ports shortlisted for onion routing detection. The traffic log file of the transfers was analysed by the IDS program and it identified correctly that no onion routing or proxy application was in use. The log file shows that it opened the FTP port (port 21) along with the randomly selected port 5167 to allow the connection to occur. If the FTP program was unlucky enough to select one of the ten listed ports, then this log would have been misidentified as having an onion routing browser application in use. The method most users will use to discover a proxy site is by using "Google" to search for the term 'proxy sites' or 'php proxy sites'. Both of these searches produce well over 12 million results each, so the next step in testing the IDS program is to establish if the program can identify the use of some of these randomly selected web-based proxy scripts revealed through the search. The IDS program was specifically designed to detect the fingerprint of three of the bigger proxy scripts available to download from the Internet; now the program will be tested to see how closely these sets of fingerprints can be used to detect unknown web-based proxy scripts.

Five web-based proxy sites were chosen at random from the selection available through the Google search. The first proxy site used the "*HTTP*" protocol along with the "*GET*" command, but for this site the "*GET*" command's detection string was not one of the defined strings in the IDS program and therefore the program misidentified the log file. The second and fifth proxy site's log files were both correctly identified as using a proxy, as the "*HTTP*" protocol was found along with the "*GET*" command and attached to the detection string

".*php?u=*". Although these sites appear different, the back-end script running the two proxy sites is the "Glype" script. Finally, the log for proxy sites three and four were analysed and correctly identified as having a proxy in use. This is because the logs matched the detection strings for a "PHP Proxy" script using the "*HTTP*" protocol, the "*GET*" command and the ".*php?q=aHR0c*" detection string. From this we can determine that from the four out of the five tests conducted for unknown proxy sites from "Google", were correctly identified as having a proxy in use, and would be flagged up to a network administrator.

6 Conclusion

The aim of this project was to research, design and develop an intrusion detection system (IDS) program, which would alert a network administrator to the use of a web-based proxy script or onion routing browser being used on their network. The program analyses a log file produced from a network protocol analyser. From the rule base established in the program, a determination is made whether an effort is being made to bypass the security measures currently in place on a network. The log file obtained for processing by the IDS program included a selection of 'normal' Internet tasks such as sending e-mails, accessing "Google" and "Wikipedia", logging into and updating social media sites and shopping on "Amazon". The reason behind gathering logs for when no proxy was being used, was to attempt to see if the IDS program could be tricked into a false reading and display a warning that a proxy or onion routing browser is being used, when it was not in use. From this perspective, the program worked exactly as it should, as not once during the rigorous testing did the program provide a false positive output message. Looking at the three web-based proxy scripts together, the IDS program was able to detect when the proxy sites were being executed to run various tasks.

The final detection that the IDS program had to deal with was to pick up the use of an onion routing browser. This was always going to be the toughest section to complete and the part of the program that stood to have the highest fail rate attached to it, as any onion routing browser is designed to make its users anonymous on the Internet—and therefore tougher to detect. So, it was not surprising to see from the results that in the original ten testing cases, the log file was incorrectly identified and instead was deemed normal Internet use. Overall, the IDS program was able to detect the use of an onion routing application and web based proxy website use by monitoring the network traffic on the client-side of a network with a 94 % detection success rate.

References

1. Gong, X., Kiyavash, N., & Borisov, N. (2010). Fingerprinting websites using remote traffic analysis. In *Proceedings of the 17th ACM conference on computer and communications* (vol. 1(1), p. 684).
2. Thomas, K., Grier, C., Ma, J., Paxson, V., & Song, D. (2011). Monarch: Providing real-time URL spam filtering as a service. In *Proceedings of the IEEE symposium on security and privacy*, Oakland, CA, USA.
3. Hamade, S. (2008). Internet filtering and censorship. In *5th International conference on information technology: New generations* (vol. 1(1), p. 1081).
4. Murdoch, S., & Anderson, R. (2008). Tools and technology of Internet filtering. *Access Denied: The Practice and Policy of Global Internet Filtering*, *1*(1), 58.
5. Caloyannides, M., Memon, N., & Venema, W. (2009). Digital forensics. *IEEE Security & Privacy*, *7*(2), 16–17.
6. Sandhu, R., & Samarati, P. (1994). Access control: Principles and practice. *IEEE Communications Magazine*, *1*(1), 40.

7. Lee, K., Jiang, Z., Kim, S., Kim, S., & Kim, S. (2005). Access control list mediation system for large-scale network. In *6th International conference on parallel and distributed computing* (vol. 1(1), p. 483).
8. Knickerbocker, P., Yu, D., & Li, J. (2009). Humboldt: A distributed phishing disruption system. In *Proceedings of the IEEE eCrime researchers summit* (pp. 1–12), Tacoma, USA.
9. Raynal, F., Ahmad, M., Shaikhli, I., & Ahmad, H. (2012). Protection of the texts using Base64 and MD5. *Journal of Advanced Computer Science and Technology Research*, *2*(1), 22.
10. Reed, M., Syverson, P., & Goldschlag, D. (1998). Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, *16*(4), 482–494.
11. Rennhard, M., Rafaeli, S., Mathy, L., Plattner, B., & Hutchison, D. (June 2002) Analysis of an anonymity network for Web browsing. In *IEEE 7th international workshop on enterprise security (WET ICE 2002)*, Pittsburgh, USA.
12. Chaabane, A., Pere Manils, P., & Kaafar, M. (2010). Digging into anonymous traffic: A deep analysis of the Tor anonymizing network. In *4th International conference on network and system security* (vol. 1(1), p. 167).
13. Reed, M., Syverson, P., & Goldschlag, D. (1996) Proxies for anonymous routing. In *12th Annual computer security applications conference* (vol. 1(1), pp. 95–102).
14. PHP Proxy. (2002). *SourceForge project*. http://sourceforge.net/projects/php-proxy.
15. Glype. (2007). *Glype proxy script*. http://www.glype.com.
16. CGI Proxy. (1996). *CGI Proxy 2.1.4*. http://www.jmarshall.com/tools/cgiproxy.
17. TOR Browser. (2002). *TOR project*. http://www.torproject.org.
18. Han, F., Chen, Z., Xu, H., & Liang, Y. (2012). A collaborative botnets suppression system based on overlay network. *International Journal of Security and Networks, 7*(4), 24–32.
19. Wireshark. (1998). *Official site*. www.wireshark.org.
20. Snort. (1998). *Official site*. http://www.snort.org.
21. Stallings, W. (1998) SSL: Web security. *Internet protocol Journal, 1*(1), 1998. http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_1-1/ssl.html.
22. Rossi, D., Mellia, M., & Meo, M. (2009). Understanding skype signalling. *The International Journal of Computer and Telecommunications Networking*, *53*(2), 130–140.

Author Biographies

Ronan O'Flaherty has an MSc in Computing and Intelligence Systems from the University of Ulster. His research interests include Internet security and detection of anonymous proxies on the Internet. He currently works as a Software Engineer in the Northern Ireland software industry.

Kevin Curran is a Reader in Computer Science at the University of Ulster and group leader for the Ambient Intelligence Research Group. His achievements include winning and managing UK & European Framework projects and Technology Transfer Schemes. Dr Curran has made significant contributions to advancing the knowledge and understanding of computer networking and systems, evidenced by over 800 published works. He is perhaps most well-known for his work on location positioning within indoor environments, pervasive computing and internet security. His expertise has been acknowledged by invitations to present his work at international conferences, overseas universities and research laboratories. He is a regular contributor to BBC radio & TV news in the UK and is currently the recipient of an Engineering and Technology Board Visiting Lectureship for Exceptional Engineers and is an IEEE Technical Expert for Internet/Security matters. He is a member of the EPSRC Peer Review College. He is the Editor in Chief of the International Journal of Ambient Computing and Intelligence and is also a member of numerous Journal Editorial boards and international conference organising committees. He has authored a number of books and is the holder of various patents.